AdaptMC: A Control-Theoretic Approach for Achieving Resilience in Mixed-Criticality Systems

Alessandro Vittorio Papadopoulos
Mälardalen University
Västerås, Sweden
alessandro.papadopoulos@mdh.se

Enrico Bini
University of Turin
Turin, Italy
bini@di.unito.it

Sanjoy Baruah
Washington University
St. Louis (MO), USA
baruah@wustl.edu

Alan Burns
University of York
York, UK
alan.burns@york.ac.uk

Abstract

A system is said to be resilient if slight deviations from expected behavior during run-time does not lead to catastrophic degradation of performance: minor deviations should result in no more than minor performance degradation. In mixed-criticality systems, such degradation should additionally be criticality-cognizant. The applicability of control theory is explored for the design of resilient run-time scheduling algorithms for mixed-criticality systems. Recent results in control theory have shown how appropriately designed controllers can provide guaranteed service to hard-real-time servers; this prior work is extended to allow for such guarantees to be made concurrently to multiple criticality-cognizant servers. The applicability of this approach is explored via several experimental simulations in a dual-criticality setting. These experiments demonstrate that our control-based run-time schedulers can be synthesized in such a manner that bounded deviations from expected behavior result in the high-criticality server suffering no performance degradation and the lower-criticality one, bounded performance degradation.

2012 ACM Subject Classification Computing methodologies → Control methods, Computer systems organization → Real-time systems

Keywords and phrases mixed criticality, control theory, run-time resilience, bounded overloads

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2018.14

Supplement Material ECRTS Artifact Evaluation approved artifact available at `https://dx.doi.org/10.4230/DARTS.4.2.1`

Funding This work was partially supported by the Swedish Foundation for Strategic Research under the project "Future factories in the cloud (FiC)" with grant number GMT14-0032. This work is also supported by NSF grants CNS 1409175, CPS 1446631, and CNS 1563845.

1 Introduction

There is an increasing trend in embedded systems towards implementing multiple functionalities upon a shared platform. It may be the case that all these functionalities are not equally important to the overall correctness of the embedded system; one widely-studied model for representing timing requirements in such systems was proposed by Vestal in a seminal paper [33]. Vestal observed that "In many applications, the consequences of missing a deadline vary in severity from task to task. In RTCA DO 178B, for example, system safety analysis assigns to each task a *criticality level* (ranging from A to D), where erroneous behavior by a level A task might cause loss of aircraft but erroneous behavior by a level D task might at worst cause inconvenient or suboptimal behavior."[1] Vestal went on to conjecture that "the higher the degree of assurance required that actual task execution times will never exceed the WCET parameters used for analysis, the larger and more conservative the latter values become in practice." (This conjecture appears reasonable. Very conservative WCET-estimation tools have been developed, typically based upon static analysis of code, that yield WCET bounds that may be very large, but that we can trust to a very high level of assurance. Less conservative WCET-estimation tools, which are typically measurement based, tend to obtain smaller estimates, but these estimates may be trust-worthy to lower levels of assurance since the worst-case behaviors of the system may not have become revealed during the measurements.) The "Vestal model" for representing, and validating the correctness of, mixed-criticality systems is based upon this conjecture. In this model,

- **§1.** A fixed number of distinct criticality levels are defined throughout the system. In this paper, we will assume that there are two such criticality levels, designated LO and HI, with the interpretation that functionalities designated as being of the LO criticality level need to have their correctness validated to a lower level of assurance than functionalities designated as being of the HI criticality level.
- **§2.** Each piece of code in the system is characterized as being of one of the criticality levels LO or HI, and by two WCET parameter estimates. One WCET estimate is determined using tools and techniques consistent with the lower criticality level LO, while the other estimate is determined using tools and techniques consistent with the higher criticality level HI.
- **§3.** Prior to run-time, the correct timing behavior (e.g., meeting deadlines) of all the functionalities are validated under the assumption that each piece of code will execute for a duration not exceeding its LO-criticality WCET estimate; in addition, the correct timing behavior of the HI-criticality functionalities (but not the LO-criticality ones) are validated under the assumption that each piece of code may execute for a duration up to its HI-criticality WCET estimate.

[1] RTCA DO 178B is a guideline dealing with the safety of safety-critical software used in certain avionics systems. Although the term "criticality" typically has a precise technical meaning in most safety standards documents, its use in [33], and subsequent use in much of the mixed-criticality scheduling theory literature, appears to be in a rather general sense as a designation of the level of assurance against failure that is desired. In this paper we are using the term in this more general sense, in keeping with prior literature in mixed-criticality scheduling.

1.1 Verification versus resilience

The Vestal approach to modeling and analysis of mixed-criticality systems, as originally proposed [33], is concerned solely with *verification* – determining, prior to run-time, whether a system will behave correctly during run-time if its run-time behavior is compliant with the models used to represent it. Clearly, such pre-runtime verification is desirable in safety-critical systems. There is an additional aspect of correctness that is also desirable: the system's run-time behavior should be *resilient* or robust in the event that run-time behavior does not conform to the models that were assumed during verification; if this happens, a robust system design ensures that performance degrades gracefully, if at all. *It is this run-time resilience aspect of system behavior that is the primary focus of this paper.* (While the precise semantics of graceful degradation should be for a particular system may depend upon the characteristics of the system, some general principles are applicable; for example, less important aspects of system functionalities should be compromised before more important ones.)

The Vestal model of [33] and its derivatives and generalizations have formed the basis of a large body of research: schedulability tests, scheduling algorithms, etc. – see, e.g., [5, 6] for a survey. Much of this research is focused upon the pre-runtime verification aspect of correctness rather than the run-time resilience. For instance, many mixed-criticality scheduling algorithms allow for LO-criticality pieces of code to be aborted if any piece of code executes beyond its LO-criticality WCET estimate. Such a scheduling algorithm may still pass the pre-runtime verification test (since such tests are only concerned with the correctness of the HI-criticality functionalities under such circumstances), but would not be considered resilient. Some recent research has attempted to provide some resilience to LO-criticality pieces of code in the event of some piece of code executing beyond its LO-criticality WCET estimate; these approaches are reviewed in Section 7.

1.2 This research

In this paper, we explore the use of control-theoretic principles to achieve resilience in mixed-criticality systems. We consider over-runs of HI-criticality pieces of code (in the sense of them executing for more than their LO-criticality WCET estimates) to be *rare events* that are best coped with by run-time adaptability. Some over-runs can be masked by under-runs by other HI-criticality pieces of code; others will require system-wide adaptation. These adaptations should be commensurate with the scale of the over-run – dropping all LO-criticality pieces of code because a single HI-criticality piece of code has executed for slightly more than its LO-criticality WCET is clearly an over-reaction. A resilient system should cope with uncertainty in a measured way.

Some recent advances in real-time control (see, e.g., [22] and the references therein) have motivated us to explore whether the desired resilience can be achieved using a control-theoretic approach. The scheduling strategy we propose has the HI-criticality workload executing within an execution-time server that is provisioned with a budget sufficient to satisfy the LO-criticality WCET requirements of this HI-criticality workload; another, similar, server is used to encapsulate the execution requirements of the LO-criticality workload. At run-time if the HI-criticality server's budget proves inadequate for meeting the execution requirements of the HI-criticality workload (due to some HI-criticality pieces of code executing for more than their LO-criticality WCET estimates) then the system is deemed to have suffered a *disturbance* or *perturbation*. We employ a *control feedback* mechanism to govern budget allocations going forward from the disturbance. This control-theoretic feedback approach allows a number of questions to be answered concerning the run-time behavior of the scheduling strategy, such as

- How long following a disturbance will it take the system to return to a non-perturbed state?
- What guaranteed level of service can be obtained for the LO-criticality workload?
- What is the maximum *magnitude* of disturbance that can be accommodated allowing for stable control and for the HI-criticality workload to remain schedulable?

1.3 Organization

The remainder of this paper is organized as follows. Section 2 presents the background for this work, while Section 3 presents AdaptMC, the proposed approach, in detail. Section 4 discusses how AdaptMC is designed and tuned, while Section 5 presents how hard real-time guarantees can be provided, by means of the calculation of the supply bound function. Section 6 presents a numerical evaluation of AdaptMC. Section 7 reviews the related work, while Section 8 concludes the paper.

2 Background

The use of feedback control to allocate resources has traditionally been applied to time-varying workloads [28, 7, 1], and the kinds of offered guarantees have been probabilistic or soft real-time. Recently, however, a control scheme called the *Self-Adaptive Server (SAS)* has been proposed [22], that provides both good average behavior and hard real-time guarantees. Such a guarantee is given by computing the supply bound function [21, 18, 27, 2] of a periodic resource supply controlled by feedback [17].

The main idea behind SAS is as follows. Each server in the system is assigned a budget of time to execute. The server is allowed execute more or less than the budget, but at the next round it will be assigned a budget that is corrected with a term that is proportional to the over- or under-run of the server. In [22] this simple, yet effective, control structure is analyzed under the assumption that the maximum over- or under-run are bounded. The designed controller is proven to effectively adapt the budget at run-time, while the supply bound function associated with the controller can be computed offline.

3 The Proposed Approach

We are concerned with mixed-criticality systems in which the LO-criticality WCET values represent typical or common-case behavior: executions *rarely* exceed these WCET values and when they do, it is typically *by small amounts*. We seek to devise resilient scheduling strategies for such mixed-criticality systems. As briefly stated in Section 1, our proposed scheduling strategy uses two servers, one each for servicing the HI-criticality and LO-criticality workloads.[2] In dimensioning these servers' budgets, our objective will be to modestly over-allocate the HI-criticality server in the sense that "most of the time" we would expect the entire provisioned budget to not be needed. If an occasional modest over-run occurs in

[2] For the kinds of application systems that we are interested in, work (in the form of "jobs") is typically generated by recurrent – periodic and sporadic – tasks; determining appropriate budget and period parameters for servers capable of accommodating the computational requirements of such recurrent tasks is an important issue that has been widely studied in the real-time scheduling community [18, 27, 2]. However, the issue of dimensioning such servers is orthogonal to the focus of this paper and we will not discuss it further, instead assuming that some appropriate scheme is used to determine appropriate server parameters such that if all jobs execute at their LO-criticality WCET estimates, then each server is able to correctly execute those jobs for which it is responsible.

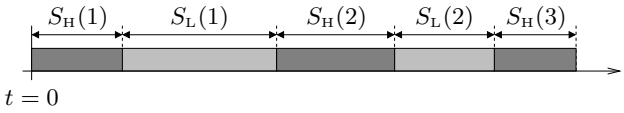

Figure 1 Server schedule over time.

the amount of execution required by the HI-criticality server (say, by an amount x over the budgeted amount), our run-time scheduling strategy is to allow the HI-criticality server to over-execute by this entire amount x, and then reduce the budget for the LO-criticality server by an amount somewhat smaller than x. Informally speaking, the hope is that after dealing with this one-time over-run, the HI-criticality server will not need to use its entire budgeted amount for some duration, and hence can compensate the LO-criticality server over this duration. However, (as we will see) our control-based scheduling strategy is robust to scenarios in which the HI-criticality server over-runs for an extended duration as well; if this happens, the LO-criticality server ends up getting under-served over an extended duration.

In order to develop a control-based strategy capable of achieving these goals, we needed to extend and adapt SAS (Self-Adaptive Server) [22] in several directions. The feedback mechanism derived in this paper is an extension of SAS to the mixed-criticality context that enables:

1. the adjustment of server budgets based on disturbances at both HI-criticality and LO-criticality servers (achieved by *cross gains* of the controller), and
2. the exploitation of the asymmetric nature of disturbances that are permitted for the LO-criticality server (which may occasionally be under-served but never receives more than its budgeted amount) to provide less conservative supply bound functions.

The presence of these two characteristics, needed in the mixed-criticality context, renders the results in [22] inapplicable directly; hence the extensions reported here. Section 3.1 below describes the adaptive scheduling strategy we have developed; the control algorithm underpinning this strategy is described in Section 3.2

3.1 Run-Time Scheduling Strategy

We propose a 2-levels hierarchical scheduler with two schedulers at the top level, one for servicing LO-criticality work and the other, for servicing HI-criticality work (see Figure 1). Let $\bar{Q}_H$ and $\bar{Q}_L$ denote the *target budgets* for the two servers, and $\bar{P} = \bar{Q}_H + \bar{Q}_L$ the *target period*. We will describe later the manner in which values are assigned to these target budget parameters; intuitively speaking, we would assign them values such that under normal circumstances (i.e., all jobs completing within their LO-criticality WCET estimates) a periodic schedule with period $\bar{P}$ in which the HI-criticality server executing for a duration $\bar{Q}_H$ is followed by the LO-criticality server executing for a duration $\bar{Q}_L$, would meet all timing requirements for all the HI-criticality and the LO-criticality workload.

During run-time these two servers are repeatedly scheduled alternately. Let us refer to the k'th time that both servers are scheduled as the *k'th round*. Let $Q_H(k)$ and $Q_L(k)$ denote the *tentative budgets* that the control algorithm computes at the end of the k'th round, for allocating to the two servers for the $(k+1)$'th round. Initially, we have $Q_H(0) = \bar{Q}_H$ and $Q_L(0) = \bar{Q}_L$; i.e., for the first round the tentative budgets are set to be equal to the target budgets.

Now suppose that during the $(k+1)$'th round for some k, the HI-criticality server needs to execute for a duration **greater** than this tentative budget $Q_H(k)$ in order to ensure the correct execution of all HI-criticality jobs (budget overrun). We allow it to do so, and let $S_H(k+1)$

denote the duration for which it executes – $S_{\text{H}}(k+1)$ is called the *actual budget* assigned to the HI-criticality server during the $(k+1)$'th round, and $\varepsilon_{\text{H}}(k) = \big(S_{\text{H}}(k+1) - Q_{\text{H}}(k)\big)$ is called the *disturbance* experienced by the HI-criticality server, i.e., the discrepancy between the target and actual budget. In response to such a disturbance, our control algorithm modifies the tentative budgets $Q_{\text{H}}(k+1)$ and $Q_{\text{L}}(k+1)$ computed for both servers for the next round, to compensate for the budget overrun and preserve the bandwidth.

3.2 The Control Algorithm

As stated earlier, our control-based scheduler is designed under the assumption that jobs executing beyond their LO-criticality WCET estimates will be rare events. The target budget $\bar{Q}_{\text{H}}$ for the HI-criticality server should be chosen to somewhat exceed the minimum needed in order to accommodate the LO-criticality WCET requirements for all the HI-criticality jobs; hence, if only one or a few jobs over-run their LO-criticality WCETs during a round, such over-runs are often masked by the excess budget and by under-runs of other HI-criticality jobs. It should only rarely be the case that such over-runs during any round get expressed as disturbances (i.e., as an $\varepsilon_{\text{H}}(k)$ value for some k); in the rare events when this does happen, our control algorithm requires that it be of magnitude that is bounded by an a priori known constant $\bar{\varepsilon}_{\text{H}}$: $|\varepsilon_{\text{H}}(k)| \leq \bar{\varepsilon}_{\text{H}}$.

In order to accommodate these disturbances in the HI-criticality servers, our control algorithm will occasionally under-schedule the LO-criticality server, providing it a supply $S_{\text{L}}(k+1)$ that is strictly less than the tentative budget $Q_{\text{L}}(k)$ that had been computed for it – when this happens, the LO-criticality server is said to experience a disturbance $\varepsilon_{\text{L}}(k) = \big(S_{\text{L}}(k+1) - Q_{\text{L}}(k)\big)$. We assume that such a disturbance will also be of magnitude that is bounded by another a priori known constant $\bar{\varepsilon}_{\text{L}}$, i.e., maximum budget over-run of the LO-criticality server.

Analogously, our run-time scheduler also bounds the "negative" disturbance to the HI-criticality server: the amount by which the actual amount of execution supplied during a round is less than the tentative budget, to have a magnitude no greater than $\bar{\varepsilon}_{\text{H}}$. Summarizing the above discussion on disturbances, we obtain the following bounds on the magnitudes of the disturbances that could be experienced by both the servers:

$$-\bar{\varepsilon}_{\text{H}} \leq \varepsilon_{\text{H}}(k) \leq \bar{\varepsilon}_{\text{H}}, \quad -\bar{\varepsilon}_{\text{L}} \leq \varepsilon_{\text{L}}(k) \leq 0. \tag{1}$$

As we had stated earlier, the actual budgets $S_{\text{H}}(k+1)$ and $S_{\text{L}}(k+1)$ assigned to the servers may be expressed as being equal to the computed tentative budgets $Q_{\text{H}}(k)$ and $Q_{\text{L}}(k)$, plus the disturbances $\varepsilon_{\text{H}}(k)$ and $\varepsilon_{\text{L}}(k)$.

$$\begin{aligned} S_{\text{H}}(k+1) &= Q_{\text{H}}(k) + \varepsilon_{\text{H}}(k) \\ S_{\text{L}}(k+1) &= Q_{\text{L}}(k) + \varepsilon_{\text{L}}(k) \end{aligned}$$

The tentative budgets $Q_{\text{H}}(k+1)$ and $Q_{\text{L}}(k+1)$ that are computed by the control algorithm may similarly be expressed as the sum of tentative budgets computed for the previous round and a corrective term (called the "*control signal*") denoted $u_{\text{H}}(k)$ and $u_{\text{L}}(k)$, that is computed by the control algorithm at the end of each round:

$$\begin{aligned} Q_{\text{H}}(k+1) &= Q_{\text{H}}(k) + u_{\text{H}}(k) \\ Q_{\text{L}}(k+1) &= Q_{\text{L}}(k) + u_{\text{L}}(k) \end{aligned}$$

Letting

$$\boldsymbol{x}(k) = \begin{bmatrix} S_{\text{H}}(k) \\ S_{\text{L}}(k) \\ Q_{\text{H}}(k) \\ Q_{\text{L}}(k) \end{bmatrix}, \quad \boldsymbol{u}(k) = \begin{bmatrix} u_{\text{H}}(k) \\ u_{\text{L}}(k) \end{bmatrix}, \quad \boldsymbol{\varepsilon}(k) = \begin{bmatrix} \varepsilon_{\text{H}}(k) \\ \varepsilon_{\text{L}}(k) \end{bmatrix},$$

one can express the *control system dynamics* – the change in values of the actual and tentative budgets across rounds that we have discussed above – in a more compact form, as follows:

$$\boldsymbol{x}(k+1) = \overbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{\boldsymbol{A}} \boldsymbol{x}(k) + \overbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}^{\boldsymbol{B}_u} \boldsymbol{u}(k) + \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}^{\boldsymbol{B}_\varepsilon} \boldsymbol{\varepsilon}(k). \tag{2}$$

We now discuss how the control signals are computed by the control algorithm (this computation is commonly referred to as the *control strategy*). In designing the controller, we assign values to four real-valued *gain* parameters $K_{\text{HH}}, K_{\text{HL}}, K_{\text{LH}}$, and K_{LL} – the parameter design is discussed in Section 4 – and compute the control signals as follows:

$$\begin{aligned} u_{\text{H}}(k) &= K_{\text{HH}}\big(\bar{Q}_{\text{H}} - S_{\text{H}}(k)\big) &&+ {}^{K_{\text{HL}}}\!/_{\gamma}\big(\bar{Q}_{\text{L}} - S_{\text{L}}(k)\big), \\ u_{\text{L}}(k) &= \gamma K_{\text{LH}}\big(\bar{Q}_{\text{H}} - S_{\text{H}}(k+1)\big) &&+ K_{\text{LL}}\big(\bar{Q}_{\text{L}} - S_{\text{L}}(k)\big). \end{aligned} \tag{3}$$

The parameters $K_{\text{HH}}, K_{\text{HL}}, K_{\text{LH}}$, and K_{LL} weigh the discrepancy between the target and actual budgets; the values assigned to these parameters reflect the effect each discrepancy has on the control signal. (Observe that in computing the control signal $u_{\text{L}}(k)$ that will be applied to the LO-criticality server, we are able to exploit the fact that the value of $S_{\text{H}}(k+1)$ is already known when the LO-criticality server is scheduled during the $(k+1)$'th round; we therefore choose to exploit this fact to compute a "better" values for $u_{\text{L}}(k)$.)

By substituting the control strategy as represented by Eqn (3) into Eqn (2), rearranging terms, and letting γ denote the ratio of the target budgets, i.e., $\gamma = \bar{Q}_{\text{L}}/\bar{Q}_{\text{H}}$, the closed-loop system dynamics may be represented as follows:

$$S_{\text{H}}(k+1) = Q_{\text{H}}(k) + \varepsilon_{\text{H}}(k) \tag{4}$$

$$S_{\text{L}}(k+1) = Q_{\text{L}}(k) + \varepsilon_{\text{L}}(k) \tag{5}$$

$$Q_{\text{H}}(k+1) = Q_{\text{H}}(k) + K_{\text{HH}}(\bar{Q}_{\text{H}} - S_{\text{H}}(k)) + {}^{K_{\text{HL}}}\!/_{\gamma}(\bar{Q}_{\text{L}} - S_{\text{L}}(k)) \tag{6}$$

$$Q_{\text{L}}(k+1) = Q_{\text{L}}(k) + K_{\text{LL}}(\bar{Q}_{\text{L}} - S_{\text{L}}(k)) + K_{\text{LH}}\gamma(\bar{Q}_{\text{H}} - S_{\text{H}}(k+1)) \tag{7}$$

or, in a more compact way:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_{CL}\,\boldsymbol{x}(k) + \boldsymbol{B}_Q\,\bar{\boldsymbol{Q}} + \boldsymbol{B}_{\varepsilon,CL}\,\boldsymbol{\varepsilon}(k) \tag{8}$$

with

$$\boldsymbol{A}_{CL} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -K_{\text{HH}} & -\frac{K_{\text{HL}}}{\gamma} & 1 & 0 \\ 0 & -K_{\text{LL}} & -\gamma K_{\text{LH}} & 1 \end{bmatrix}, \quad \boldsymbol{B}_Q = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ K_{\text{HH}} & \frac{K_{\text{HL}}}{\gamma} \\ \gamma K_{\text{LH}} & K_{\text{LL}} \end{bmatrix},$$

$$\bar{\boldsymbol{Q}} = \begin{bmatrix} \bar{Q}_{\text{H}} \\ \bar{Q}_{\text{L}} \end{bmatrix}, \quad \boldsymbol{B}_{\varepsilon,CL} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ -\gamma K_{\text{LH}} & 0 \end{bmatrix}.$$

The eigenvalues of $\boldsymbol{A}_{CL}$ determine the convergence time towards the value $\overline{\boldsymbol{x}}$ for the system state. These can be obtained from the characteristic polynomial of $\boldsymbol{A}_{CL}$:

$$p(z) = z^4 - 2z^3 + (K_{\text{HH}} + K_{\text{LL}} + 1)\, z^2 - (K_{\text{HH}} + K_{\text{LL}} + K_{\text{HL}}K_{\text{LH}})z + K_{\text{HH}}K_{\text{LL}}. \tag{9}$$

Since the considered system is linear, we can use the superposition principle[3], and consider separately the effect of $\bar{\boldsymbol{Q}}$ and $\boldsymbol{\varepsilon}$ on the evolution of $\boldsymbol{x}$. The z-transform of (8) is:

$$\boldsymbol{X}(z) = (z\boldsymbol{I} - \boldsymbol{A}_{CL})^{-1}\left(\boldsymbol{x}(0) + \boldsymbol{B}_Q\bar{\boldsymbol{Q}} + \boldsymbol{B}_{\varepsilon,CL}\boldsymbol{E}(z)\right) \tag{10}$$

Evaluating the transfer function from the error ε to the state x for $z = 1$ computes, in control theoretical terms, the asymptotic effect of the unitary constant disturbance ε on the state x; in the considered case, evaluating $(z\boldsymbol{I} - \boldsymbol{A}_{CL})^{-1}\boldsymbol{B}_{\varepsilon,CL}$ for $z = 1$ yields:

$$(\boldsymbol{I} - \boldsymbol{A}_{CL})^{-1}\boldsymbol{B}_{\varepsilon,CL} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$$

that proves that the effect of $\boldsymbol{\varepsilon}$ on $\boldsymbol{S}$ (the first two rows) vanishes asymptotically to zero independently of the values assigned to the gain parameters. The effect of a unitary constant disturbance on the budgets $\boldsymbol{Q}$, on the other hand, is to compensate $\boldsymbol{\varepsilon}$ by reducing the budget of exactly a unity so that value of $\boldsymbol{S}$ will compensate perfectly the disturbance $\boldsymbol{\varepsilon}$.

4 Designing the Control Algorithm

In Section 3 we described how the control logic can be used to adjust the resource budgets allocated to HI and LO-criticality servers. In this section, we are going to explore the assignment of values to the control gain parameters K_{HH}, K_{HL}, K_{LH}, and K_{LL} such that the resulting budget dynamics are guaranteed to possess the desirable control-theoretic properties of *compensation* and *stability*.

▶ **Definition 1** (Compensation property)**.** A single disturbance $\varepsilon(k)$ on the HI/LO-criticality server results in an opposite or null effect on the value of $S(k+1)$ (i.e., the actual budget) of the LO/HI-criticality server, i.e.,

$$\exists n > 0 : \varepsilon_i(k) = -\alpha(k+n)u_j(k+n), \quad \alpha(k+n) \geq 0, i, j \in \{\text{H}, \text{L}\}, \text{ and } i \neq j.$$

The intuition of the compensation property is that whenever the HI-criticality server exceeds its budget ($S_{\text{H}}(k+1) > Q_{\text{H}}(k)$), the LO-criticality server compensates for this disturbance by temporarily reducing its budget. On the other hand, when the LO-criticality server requires less time for its execution ($S_{\text{L}}(k+1) < Q_{\text{L}}(k)$), then the HI-criticality server will be allowed to temporarily increase its budget. Finally, when the HI-criticality server executes for less time ($S_{\text{H}}(k+1) < Q_{\text{H}}(k)$), then the LO-criticality server can temporarily increase its budget.

The overall objective is to both preserve the bandwidth of the two servers, and to reach the target period $\bar{P} = \bar{Q}_H + \bar{Q}_L$.

[3] The *superposition principle* for linear systems states that the net response caused by multiple stimuli upon such a system is equal to the sum of the responses that would have been caused by each individual stimulus.

▶ **Theorem 2.** *If*

$$K_{\text{HH}} > 0,\ K_{\text{HL}} \geq 0,\ K_{\text{LH}} \geq 0,\ K_{\text{LL}} > 0, \tag{11}$$

then the system (8) *exhibits the compensation property.*

Proof. First, let us consider the case when $K_{ii} > 0$, $K_{\text{HL}} = K_{\text{LH}} = 0$, $i \in \{\text{H}, \text{L}\}$ makes the HI- and LO-criticality systems completely decoupled. It is trivial to show that the compensation property holds, since ε_{H} has no effect on the LO-criticality server, and ε_{L} has no effect on the HI-criticality server.

Therefore, we focus on the case $K_{ij} > 0$, $i, j \in \{\text{H}, \text{L}\}$. Since we are dealing with a linear system, we can consider the effect of the disturbances separately, and then use the superposition principle. Without loss of generality, let us consider a positive disturbance $\varepsilon_{\text{H}} > 0$, and an initial condition $S_i(0) = Q_i(0) = \bar{Q}_i$, $i \in \{\text{H}, \text{L}\}$. First, consider the case when $K_{ij} > 0$, $i, j \in \{\text{H}, \text{L}\}$. ε_{H} has the effect of increasing the value of S_{H}, according to (4), without affecting immediately the value of S_{L}, according to (5). If $K_{ij} > 0$, $i, j \in \{\text{H}, \text{L}\}$, an increasing value of S_{H} will make decrease both the tentative budgets, as per (6), and (7). Therefore, in the next step, the tentative budget allocated to the two servers is decreased, with the effect that S_{H} is above the desired budget $\bar{Q}_{\text{H}}$, while S_{L} is below the desired budget $\bar{Q}_{\text{L}}$.

Analogous considerations can be done for the respective negative case. This concludes the proof. ◀

Notice that the compensation property of the control scheme of (8) relates to the transient behavior caused by the occurrence of a disturbance – it does not guarantee that the effect of a disturbance will eventually vanish. Hence a second essential property of the control scheme of (8) is *stability*. If stability is not guaranteed, then it is not possible to preserve the bandwidth, and not even to preserve the target period $\bar{P}$. We want the effect of transient perturbations to be transient, and desire that the actual server budgets tend towards the specified target budget values. Theorem 2 guarantees some properties on the initial transient, but it does not guarantee the convergence of the system behavior towards the desired budget; guaranteeing such convergence is equivalent, in control theory terminology, to requiring stability of the controlled system.

Stability of discrete-time systems, such as the one specified by Expression (8), is guaranteed if and only if the roots of the characteristic polynomial $p(z)$ of (9) are within the unit circle over the complex plane $\mathbb{C}$. That is

$$p(z) = 0 \quad \Rightarrow \quad |z| < 1.$$

Such a condition on the polynomial $p(z)$ can be translated into a condition over the coefficients of the polynomial and, in turn, into a condition over the control gains K_{HH}, K_{HL}, K_{LH}, and K_{LL}. Jury's stability criterion (see, for example, [23, Sec 3.15.2]) offers a necessary and sufficient condition for the stability of a discrete-time system in the form of a set of inequalities which are functions of the coefficients of the characteristic polynomial. By applying Jury's criterion to the polynomial $p(z)$ of (9), one can obtain four analytic conditions on the values of the parameters K_{ij}, $i, j \in \{\text{H}, \text{L}\}$ that guarantee stability. We do not present these conditions here since they are quite lengthy and complex, but point out that they can be computed through a symbolic manipulation tool[4] from the expression of $p(z)$.

[4] We used the Matlab function available at `https://se.mathworks.com/matlabcentral/fileexchange/13904-jury` in combination with the Matlab symbolic toolbox.

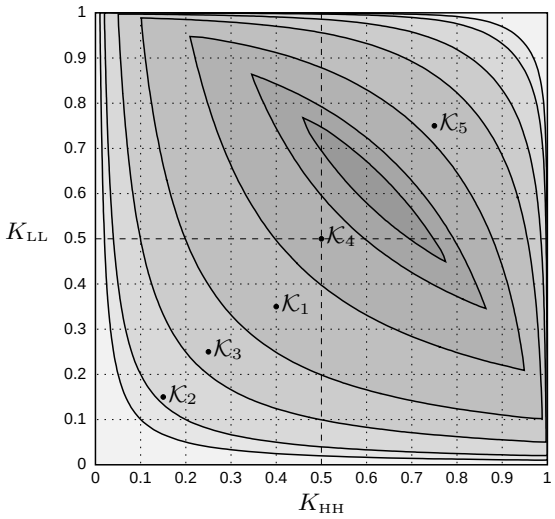

Figure 2 Region of feasible control gains. The illustrated regions correspond to the values of $K_{\mathrm{i}} \in \{0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.35\}$, respectively from the larger region to the smaller one. Black dots represent the gains of the controllers selected for the examples illustrated in Section 6.

The intersection of the inequalities (11) with the stability conditions that are obtained with the Jury criterion describes the region of the feasible controller gains that guarantee both the compensation property and the stability of the controller. Figure 2 shows the contour plot of the stability regions for the parameters K_{HH}, K_{LL}, for different values of $K_{\mathrm{i}} = K_{\mathrm{HL}}K_{\mathrm{LH}}$ (identified in the figure with different colors). Notice that the region is symmetric with respect to the plane $K_{\mathrm{HH}} = K_{\mathrm{LL}}$, and that for increasing K_{i} the stability region shrinks. Moreover, for $K_{\mathrm{i}} = 0$, the stability region is $0 < K_{\mathrm{HH}} < 1$, and $0 < K_{\mathrm{LL}} < 1$.

5 Bounding the Resource Supply

Feedback control for real-time resource allocation was initially used for tracking time-varying workloads [28, 7, 1]. Because of the unpredictable nature of variations, the type of offered guarantees are probabilistic or soft real-time. Recently, however, it was shown that a control scheme can provide both a good average behavior *and* hard-real-time guarantees [22]. Such a guarantee was given by computing the "supply bound function" of a periodic resource supply controlled by a feedback loop such as the one described by Expression (8).

Bounds to supply functions are a commonly used abstraction for modeling the minimum amount of a computing resource that is available over time [21, 18, 27, 2]. They have demonstrated their applicability to realistic use cases (e.g., avionics [12]) and there exist measurement-based tools to determine them from actual system execution traces [20]. Let us briefly recall the main concepts. Let $s(t)$ be the indicator function of the availability of a resource:

$$s(t) = \begin{cases} 1 & \text{the resource is available at time } t \\ 0 & \text{the resource is not available at time } t, \end{cases} \tag{12}$$

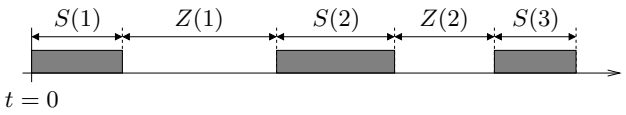

Figure 3 Active intervals interleaved with idle intervals.

Then the *supply bound function* $\mathsf{sbf}(t)$ is such that it is

$$\forall t_0, t, \quad \mathsf{sbf}(t) \leq \int_{t_0}^{t_0+t} s(\tau)\, d\tau. \tag{13}$$

Clearly, from (13), the bound $\mathsf{sbf}(t)$ may not be unique. The aim of much of the research in this area is to find valid bounds $\mathsf{sbf}(t)$ fulfilling (13), which are as high as possible.

In [22], the resource availability schedule is modeled as a sequence of *active* intervals of duration $S(k)$ in which the resource is provided, alternating with intervals of *idle* time of duration $Z(k)$. An example of such a schedule and the corresponding representation by means of the sequences $S(k)$ and $Z(k)$ is illustrated in Figure 3. Such a model offers some advantages over the traditional model by the indicator function of a schedule (as in Eq. 12). In fact, it was proved (Lemma 1 in [22]) that the supply function lower bound $\mathsf{sbf}(t)$ can be written as a function of the sequences of active and idle intervals. Specifically, it was shown that if the resource offered by a schedule is modeled by a sequence of supply intervals of length $\{S(k)\}_{k=1,2,\ldots}$ interleaved by a sequence of idle intervals of length $\{Z(k)\}_{k=1,2,\ldots}$, then the following constitutes a valid supply bound function for this resource availability:

$$\mathsf{sbf}(t) = \min\left\{t - \sigma_Z(n), \sigma_S(n)\right\}, \quad t \in \mathbb{I}_n, n \in \mathbb{N} \tag{14}$$

with the sequence of intervals $\{\mathbb{I}_n\}_{n\in\mathbb{N}}$ defined as

$$\mathbb{I}_n = \begin{cases} \big[0, \sigma_Z(1)\big) & n = 0 \\ \big[\sigma_Z(n) + \sigma_S(n-1), \sigma_Z(n+1) + \sigma_S(n)\big) & n \geq 1 \end{cases} \tag{15}$$

and with

$$\sigma_S(n) = \inf_{n_0} \sum_{k=n_0}^{n_0+n-1} S(k), \quad \sigma_Z(n) \quad = \sup_{n_0} \sum_{k=n_0}^{n_0+n-1} Z(k), \tag{16}$$

properly extended at $n = 0$ with $\sigma_S(0) = \sigma_Z(0) = 0$. The worst-case nature of the bound is condensed in $\sigma_S(n)$ that is the smallest sum of the lengths of n consecutive active intervals (respectively, $\sigma_Z(n)$ is the largest sum of the length of n consecutive idle intervals). Figure 4 illustrates an example of supply function $\mathsf{sbf}(t)$. In the figure, we also draw on top the extent of the intervals $\mathbb{I}_n$.

5.1 Characterizing the Server Supply Functions

One criticism of many mixed-criticality scheduling algorithms that have been proposed is that the LO-criticality workload is severely penalized (e.g., dropped entirely) in the event of the mixed-criticality system behavior exceeding its LO-criticality specifications. As stated earlier, this violates the principle of resilience or robustness, which requires that slight deviations from LO-criticality specifications should result in slight degradation of performance (in mixed-criticality settings, to only the LO-criticality workload). In this section, we discuss

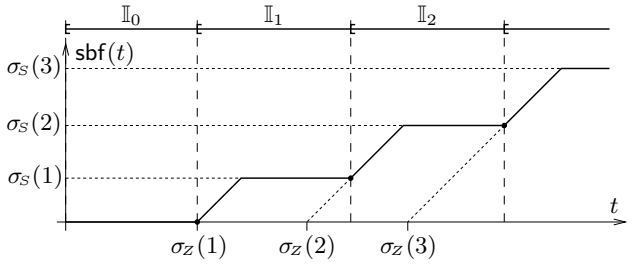

Figure 4 An example of supply bound function $\mathsf{sbf}(t)$ for a resource supply described by sequences $S(k)$ and $Z(k)$ of active and idle intervals.

how an appropriate assignment of values to the gains of the controller $K_{\text{HH}}, K_{\text{HL}}, K_{\text{LH}}$, and K_{LL} enables such resilience by guaranteeing some resource supply to the LO-criticality server.

Our overall approach is inspired by, and based upon, the analysis proposed by Papadopoulos et al. [22]. However, there are several differences in the server requirements/assumptions between our model and the model in [22], that renders the main result (Theorem 1 of [22, page 231]) inapplicable for our purposes.

- First, while disturbances were assumed in [22] to have symmetric bounds, in this paper the LO-criticality server may only experience a a *negative* disturbance, as in (1); equivalently, the LO-criticality server is never allowed to execute beyond the tentative budget that is computed for it by the control strategy.
- Second, in our mixed-criticality run-time algorithm, the servers assigning the computing resource are *coupled* by cross gains K_{HL} and K_{LH}: letting $i, j \in \{\text{H}, \text{L}\}$, it is possible to correct the server budget $S_i(k+1)$ based on any disturbance $\varepsilon_j(k)$. This enables a more prompt compensation.

The following theorem characterizes the relationship between the run-time behavior of the two servers, and enables us to determine the supply function of both the HI-criticality and LO-criticality servers. In the theorem we use the notation $h_{ij}(k)$, $g_{ij}(k)$, and $r_{ij}(k)$ to denote the *impulse*, *step*, and *ramp* responses, respectively, of the system with input $\varepsilon_j(k)$ and output $S_i(k)$, with $i, j \in \{\text{H}, \text{L}\}$ (see Appendix A for the definitions of the considered input signals).

▶ **Theorem 3.** *Consider a pair of* HI*-criticality and* LO*-criticality servers, whose budgets $S_{\text{H}}(k)$ and $S_{\text{L}}(k)$ are subject to disturbances $\varepsilon_{\text{H}}(k)$ and $\varepsilon_{\text{L}}(k)$ respectively, with closed-loop system dynamics as specified by Equation (8). If the disturbances are bounded as specified by (1), then the supply function $\mathsf{sbf}_{\text{H}}(t)$ of the* HI*-criticality server is as specified in Equation (14) with*

$$\begin{aligned} \sigma_S(n) &= n\bar{Q}_{\text{H}} - \bar{\varepsilon}_{\text{H}}\mathcal{N}_{\text{HH}}(n) - \frac{\bar{\varepsilon}_{\text{L}}}{2}\big(\mathcal{I}_{\text{HL}}(n) + \mathcal{N}_{\text{HL}}(n)\big), \\ \sigma_Z(n) &= n\bar{Q}_{\text{L}} + \bar{\varepsilon}_{\text{H}}\mathcal{N}_{\text{LH}}(n) + \frac{\bar{\varepsilon}_{\text{L}}}{2}\big(\mathcal{J}_{\text{LL}}(n) + \mathcal{N}_{\text{LL}}(n)\big), \end{aligned} \tag{17}$$

and the supply function $\mathsf{sbf}_{\text{L}}(t)$ of the LO*-criticality server is as specified in Equation (14) with*

$$\begin{aligned} \sigma_S(n) &= n\bar{Q}_{\text{L}} - \bar{\varepsilon}_{\text{H}}\mathcal{N}_{\text{LH}}(n) - \frac{\bar{\varepsilon}_{\text{L}}}{2}\big(\mathcal{I}_{\text{LL}}(n) + \mathcal{N}_{\text{LL}}(n)\big), \\ \sigma_Z(n) &= n\bar{Q}_{\text{H}} + \bar{\varepsilon}_{\text{H}}\mathcal{N}_{\text{HH}}(n) + \frac{\bar{\varepsilon}_{\text{L}}}{2}\big(\mathcal{J}_{\text{HL}}(n) + \mathcal{N}_{\text{HL}}(n)\big). \end{aligned} \tag{18}$$

The coefficients $\mathcal{N}_{ij}(n)$, $\mathcal{I}_{i\text{L}}(n)$, *and* $\mathcal{J}_{i\text{L}}(n)$ *used in the equations above are set as*

$$\begin{aligned}\mathcal{N}_{ij}(n) &= \sum_{k=0}^{\infty} \left| g_{ij}(k) - g_{ij}(k-n) \right| \\ \mathcal{I}_{i\text{L}}(n) &= \sup_k \left\{ r_{i\text{L}}(k) - r_{i\text{L}}(k-n) \right\} \\ \mathcal{J}_{i\text{L}}(n) &= \sup_k \left\{ r_{i\text{L}}(k-n) - r_{i\text{L}}(k) \right\}\end{aligned} \tag{19}$$

with $i, j \in \{\text{H}, \text{L}\}$ *corresponding to the* LO*-criticality and* HI*-criticality servers, respectively.*

Proof. In the appendix (Appendix A). ◀

Theorem 3 enables us to determine the supply function of both the HI-criticality and LO-criticality servers. In the next section, several design choices for the control gain parameters are illustrated and discussed; it is shown how different desired behaviors can be achieved by an appropriate choice of gain parameters.

6 Evaluation via Simulation

By characterizing the run-time dynamics of both the HI-criticality and the LO-criticality server, Equation (8) and Theorem 3 allow us to estimate the system response to different kinds of transient deviations from the expected "common-case" behavior, as characterized by the LO-criticality WCET estimates. We now explore, via some simulation experiments, (i) the manner in which the choice of gain parameter values influences the precise nature of resilience exhibited by the run-time scheduler, and (ii) how our proposed scheme compares with a simpler alternative strategy that is not based on the application of control-theoretic principles.

6.1 The Influence of Parameter Values

A closed-form solution of the dynamics of the system (8) may be obtained with the Lagrange formula for the solution of a set of linear difference equations (see, e.g., [23, Section 12.3.5, Eq. (12.3-34a)] for a text-book discussion). We consider the following set of parameters that are expressed as $\mathcal{K}_i = \{K_{\text{HH}}, K_{\text{HL}}, K_{\text{LH}}, K_{\text{LL}}\}$:

$$\mathcal{K}_1 = \{0.4, 0.1, 0.1, 0.35\}, \quad \mathcal{K}_2 = \{0.15, 0.1, 0.1, 0.15\}, \quad \mathcal{K}_3 = \{0.25, 0.1, 0.1, 0.25\},$$
$$\mathcal{K}_4 = \{0.5, 0.1, 0.1, 0.5\}, \quad \mathcal{K}_5 = \{0.75, 0.1, 0.1, 0.75\}$$

Notice that all the selected sets of parameters satisfy the stability conditions, and the compensation property conditions, and therefore lie in the region as depicted in Figure 2.

We considered the case of the following target budgets: $\bar{Q}_{\text{H}} = 10$, $\bar{Q}_{\text{L}} = 8$, i.e., $\gamma = 0.8$, and $\varepsilon_{\text{H}} = 1$, $\varepsilon_{\text{L}} = 1$. The resulting supply functions are presented in Figure 5. One can see that the supply function associated with $\mathcal{K}_1$ is higher than the others.

If keeping with common practice in control theory, we also analyzed the controller response to a *constant* disturbance. Figure 6 shows the effect of the disturbance while varying the values of K_{ij}, $i, j \in \{\text{H}, \text{L}\}$. From Figure 6 we conclude that the best value for the parameters is $\mathcal{K}_1$, since it provides a faster convergence to the target budget, and with negligible oscillations.

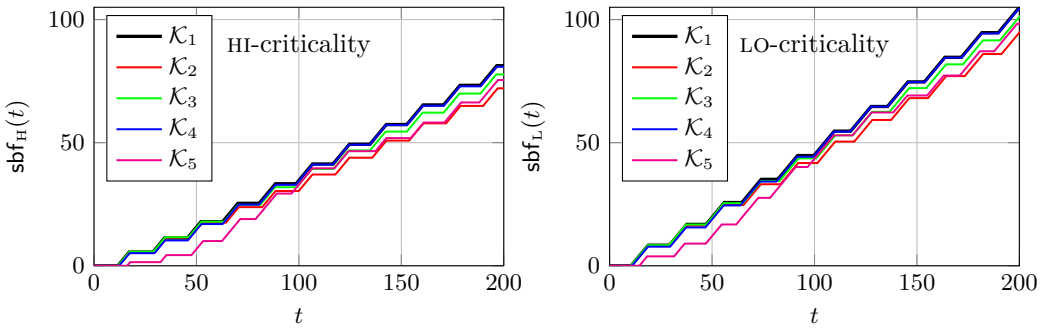

Figure 5 Supply functions for the considered set of control parameters.

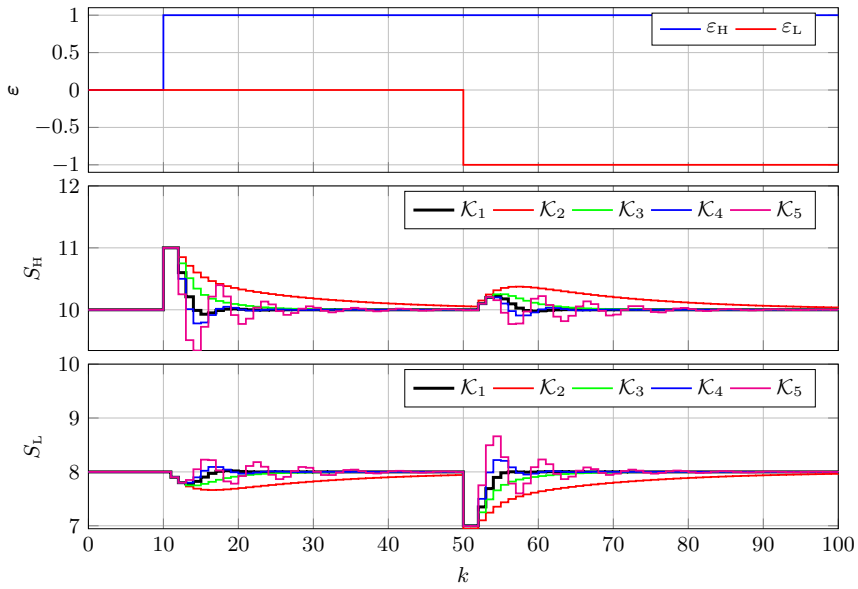

Figure 6 Effect of constant disturbances with various selection of K_{ij}, $i, j \in \{\mathrm{H}, \mathrm{L}\}$.

6.2 Comparison with an Alternative Scheme

We now compare the presented approach with a *Period-Preserving Approach (PPA)*, described next. Based upon the findings described in Section 6.1 above, in these experiments we have selected the parameter values $\mathcal{K}_1$ for AdaptMC.

In the PPA the HI-criticality and LO-criticality servers execute in sequence and periodically, with a fixed period P (equal to the target period for AdaptMC). Within each period, the HI-criticality server executes as much as it needs, allowing for any overrun, and the remaining budget of the period is allocated to the LO-criticality server. Formally, with the introduced notation:

$$S_{\mathrm{H}}(k+1) = Q_{\mathrm{H}}(k) + \varepsilon_{\mathrm{H}}(k)$$
$$S_{\mathrm{L}}(k) = P - S_{\mathrm{H}}(k+1)$$

where P now is a fixed value. PPA represents the simplest and most intuitive way to compensate for non-ideal executions of the HI-criticality server.

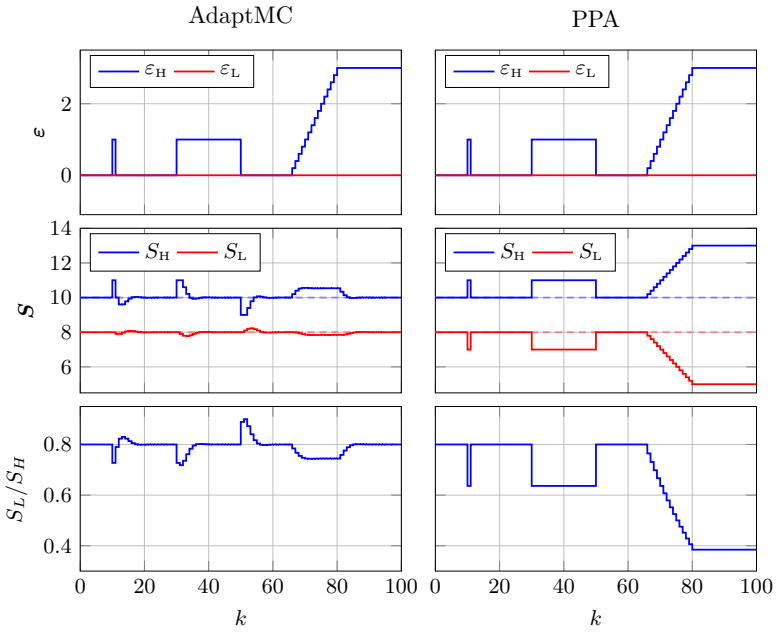

Figure 7 Comparison between AdaptMC and PPA.

In order to present the main differences between AdaptMC and PPA, we consider a scenario in which three types of disturbances occur in the system: impulse, constant, and linearly increasing. (In a well-designed mixed-criticality system, the most common form of deviation from expected behavior should be of the kind best modeled as an *impulse* disturbance – an overload that lasts for just one round and occurs rarely enough that the effect of one such overload will have completely dissipated by the time the next one occurs.)

The system is initialized as $S_{\text{H}}(0) = Q_{\text{H}}(0) = \bar{Q}_{\text{H}} = 10$, and $S_{\text{L}}(0) = Q_{\text{L}}(0) = \bar{Q}_{\text{L}} = 8$, $P = 18$ and no disturbance ε is present. An impulse overrun occurs at round 10, a constant overrun occurs between rounds 30 and 50, and a linearly increasing disturbance begins at round 65, and increases until it becomes of magnitude $\bar{\varepsilon}_{\text{H}}$. Figure 7 summarizes the obtained numerical results. The graphs in the first row show the time evolution of the HI-criticality server overruns: this is the disturbance, and is the same for the AdaptMC and PPA. The graphs in the second row compares the actual time executed by the two servers with the two methods. AdaptMC reacts to the disturbances by trying to preserve the target budgets, and making minor adjustments to the tentative budgets. PPA, on the other hand, favors the overruns of the HI-criticality server, while the execution of the LO-criticality server is severely affected. Finally, the last row of Figure 7 shows the ratio between the bandwidth allocated for the LO-criticality server, i.e., S_{L}/P, and the actual bandwidth allocated for the HI-criticality server, i.e., S_{H}/P. We call this, the *bandwidth ratio*, and it is defined as: $S_{\text{L}}/S_{\text{H}}$. The target bandwidth is $\bar{Q}_{\text{L}}/P = 8/18$, and $\bar{Q}_{\text{H}}/P = 10/18$, i.e., the target bandwidth ratio is $\bar{Q}_{\text{L}}/\bar{Q}_{\text{H}} = 8/10$. The average bandwidth ratio allocated with AdaptMC is much closer to the target bandwidth ratio than with PPA , and even the maximum deviation from the target bandwidth is minimized by AdaptMC thanks to the feedback scheme.

7 Related Work

The key property of the control-theoretic approach to budget control described in this paper is the dynamic manner in which it modifies budgets to deal with different sizes and types of task overruns; this stands in sharp contrast to the approach adopted in most other scheduling schemes for mixed-criticality systems. In these schemes during run-time the system is defined to be in one of two modes of behaviors. In the LO-criticality or "normal" mode all tasks are executing within their LO-criticality WCET estimates and all deadlines (of both HI- and LO-criticality tasks) are being met. As soon as any HI-criticality task executes for more than its LO-criticality WCET estimate then there is a system-wide mode change to the HI-criticality mode. In this new mode the behavior of the system is quite different. The change to the HI-criticality mode occurs even if a single HI-criticality task executes for a miniscule amount more than its LO-criticality WCET estimate or, at the other extreme, if all HI-criticality tasks execute at their HI-criticality WCET estimate. The system responds in the same way: there is no attempt to define behaviors that are commensurate with the magnitude of the overrun (the disturbance or perturbation as defined in this paper).

Following a criticality mode change there are a number of approaches that have been developed to define the degraded behavior of the system in the HI-criticality mode. The most extreme is to just implement the assumptions made during the verification of the system. Here, in the HI-criticality mode, only the HI-criticality tasks are guaranteed; hence all the LO-criticality tasks can be abandoned (aborted). This is clearly an unacceptable approach as no attempt is made to survive the overrun; there is no resilience in the run-time behavior of the system. Forms of resilience that have been developed include:

1. Reduce priorities of the LO-criticality tasks [3], or similar with EDF scheduling [13].
2. Increase the periods and deadlines of LO-criticality jobs [32, 31, 15, 30, 29, 25], called *task stretching*, the *elastic task model* or *multi-rate*.
3. Impose only a weakly-hard constraint on the LO-criticality jobs [9].
4. Decrease the computation times of some or all of the LO-criticality tasks [4], perhaps by utilizing an imprecise mixed-criticality (IMC) model [19, 24] or budget control [10].
5. Abandon LO-criticality work in a disciplined sequence [8, 14, 11, 26, 16].

A flexible scheme utilizing hierarchical scheduling is proposed by Gu et al. [10]. They differentiate between minor violations of LO-criticality execution time which can be dealt with within a component (an internal mode change) and more extensive violations that requires a system-wide external mode change.

By removing entirely the notion of a mode change (and hence a single perhaps quite severe change in system behavior), the approach proposed in this paper results in more gradual and measured responses to rare temporal glitches, such responses being automatically delivered by the developed feedback scheme.

8 Conclusions and Future Work

In this paper we have shown how a control-theoretic approach based upon servers can be used to manage the budgets allocated to dual-criticality workloads. The control strategy developed automatically responds to minor perturbations in the needs of the HI-criticality server with minimum and bounded degradation in the service provided to the LO-criticality server. The controller is defined by four "*gain*" parameters whose values must be constrained in order to ensure stable and appropriate (compensated) control; nevertheless there remains considerable freedom for the designer to tune the behavior of the controller. This has been demonstrated by some simple examples.

This initial study has been limited to just two criticality levels and two servers (one per level). Future work will first look to increase the number of levels supported, and to investigate if there is any benefit to be gained from having more than one HI-criticality server (and more than one LO-criticality server).

References

1. Luca Abeni, Luigi Palopoli, Giuseppe Lipari, and Jonathan Walpole. Analysis of a reservation-based feedback scheduler. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium*, pages 71–80, Austix (TX), USA, dec 2002.
2. Luis Almeida and Paulo Pedreiras. Scheduling within temporal partitions: response-time analysis and server design. In *Proceedings of the* 4^{th} *ACM International Conference on Embedded Software*, pages 95–103, Pisa, Italy, 2004.
3. Sanjoy K. Baruah and Alan Burns. Implementing mixed criticality systems in Ada. In A. Romanovsky, editor, *Proc. of Reliable Software Technologies - Ada-Europe 2011*, pages 174–188. Springer, 2011.
4. Alan Burns and Sanjoy K. Baruah. Towards a more practical model for mixed criticality systems. In *Proc. 1st Workshop on Mixed Criticality Systems (WMC), RTSS*, pages 1–6, 2013.
5. Alan Burns and Robert I. Davis. A survey of research into mixed criticality systems. *ACM Computer Surveys*, 50(6):1–37, 2017.
6. Alan Burns and Robert I. Davis. Mixed-criticality systems: A review (10th edition). (Accessed on Apr 8th, 2018), 2018. URL: `http://www-users.cs.york.ac.uk/~burns/review.pdf`.
7. Anton Cervin and Johan Eker. Feedback scheduling of control tasks. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 4871–4876, 2000. `doi:10.1109/CDC.2001.914702`.
8. Tom Fleming and Alan Burns. Incorporating the notion of importance into mixed criticality systems. In Liliana Cucu-Grosjean and Robert I. Davis, editors, *Proc. 2nd Workshop on Mixed Criticality Systems (WMC), RTSS*, pages 33–38, 2014.
9. Oliver Gettings, Sophie Quinton, and Robert I. Davis. Mixed criticality systems with weakly-hard constraints. In *Proc. 23rd International Conference on Real-Time Networks and Systems (RTNS 2015)*, pages 237–246, 2015.
10. Xiaozhe Gu and Arvind Easwaran. Dynamic budget management with service guarantees for mixed-criticality systems. In *Proc. Real-Time Systems Symposium (RTSS)*, pages 47–56. IEEE, 2016.
11. Xiaozhe Gu, Arvind Easwaran, Kieu-My Phan, and Insik Shin. Resource efficient isolation mechanisms in mixed-criticality scheduling. In *Proc. 27th ECRTS*, pages 13–24. IEEE, 2015.
12. Ana Guasque, Patricia Balbastre, and Alfons Crespo. Real-time hierarchical systems with arbitrary scheduling at global level. *Journal of Systems and Software*, 119:70–86, 2016.
13. Pengcheng Huang, Georgia Giannopoulou, Nikolay Stoimenov, and Lothar Thiele. Service adaptions for mixed-criticality systems. In *Proc. 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Singapore, 2014.
14. Pengcheng Huang, Pratyush Kumar, Nikolay Stoimenov, and Lothar Thiele. Interference constraint graph – a new specification for mixed-criticality systems. In *Proc. 18th Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2013.
15. Mathieu Jan, Lilia Zaourar, and Maurice Pitel. Maximizing the execution rate of low criticality tasks in mixed criticality system. In *Proc. 1st WMC, RTSS*, pages 43–48, 2013.

16 Jaewoo Lee, Hoon Sung Chwa, Linh T. X. Phan, Insik Shin, and Insup Lee. MC-ADAPT: Adaptive task dropping in mixed-criticality scheduling. *ACM Trans. Embed. Comput. Syst.*, 16:163:1–163:21, 2017.

17 Alberto Leva and Martina Maggio. Feedback process scheduling with simple discrete-time control structures. *IET control theory & applications*, 4(11):2331–2342, 2010.

18 Giuseppe Lipari and Enrico Bini. Resource partitioning among real-time applications. In *Proceedings of the 15-th Euromicro Conference on Real-Time Systems*, pages 151–158, Porto, Portugal, 2003.

19 D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov, and W. Yi. EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees. In *Proc. IEEE RTSS*, pages 35–46, 2016.

20 Martina Maggio, Juri Lelli, and Enrico Bini. A tool for measuring supply functions of execution platforms. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2016 IEEE 22nd International Conference on*, pages 39–48. IEEE, 2016.

21 Aloysius K. Mok, Xiang Feng, and Deji Chen. Resource partition for real-time systems. In *Proceedings of the* 7^{th} *IEEE Real-Time Technology and Applications Symposium*, pages 75–84, Taipei, Taiwan, 2001.

22 Alessandro Vittorio Papadopoulos, Martina Maggio, Alberto Leva, and Enrico Bini. Hard real-time guarantees in feedback-based resource reservations. *Real-Time Systems*, 51(3):221–246, 2015.

23 Paraskevas N. Paraskevopoulos. *Modern Control Engineering*. Automation and Control Engineering. Taylor & Francis, 2001.

24 Risat Mahmud Pathan. Improving the quality-of-service for scheduling mixed-criticality systems on multiprocessors. In Marko Bertogna, editor, *Proc. Euromicro Conference on Real-Time Systems (ECRTS)*, volume 76 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

25 Saravanan Ramanathan, Arvind Easwaran, and Hyeonjoong Cho. Multi-rate fluid scheduling of mixed-criticality systems on multiprocessors. *Real-Time Systems*, Online First, 2017.

26 Jiankang Ren and Linh Thi Xuan Phan. Mixed-criticality scheduling on multiprocessors using task grouping. In *Proc. 27th ECRTS*, pages 25–36. IEEE, 2015.

27 Insik Shin and Insup Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of the* 24^{th} *Real-Time Systems Symposium*, pages 2–13, Cancun, Mexico, dec 2003.

28 John A. Stankovic, Chenyang Lu, Sang H. Son, and Gang Tao. The case for feedback control in real-time scheduling. In *Proceedings of the Euromicro Conference on Real-Time*, York, U.K., jun 1999.

29 Hang Su, Peng Deng, Dakai Zhu, and Qi Zhu. Fixed-priority dual-rate mixed-criticality systems: Schedulability analysis and performance optimization. In *Proc. Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 59–68. IEEE, 2016.

30 Hang Su, Nan Guan, and Dakai Zhu. Service guarantee exploration for mixed-criticality systems. In *Proc. Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–10. IEEE, 2014.

31 Hang Su and Dakai Zhu. An elastic mixed-criticality task model and its scheduling algorithm. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE, pages 147–152, 2013.

32 Hang Su, Dakai Zhu, and Daniel Mosse. Scheduling algorithms for elastic mixed-criticality tasks in multicore systems. In *Proc. RTCSA*, 2013.

33 Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the Real-Time Systems Symposium*, pages 239–243, Tucson, AZ, December 2007. IEEE Computer Society Press.

A Proof of Theorem 3

Before entering the details of the proofs, we remind that a linear time-invariant (LTI) system can be uniquely characterized by its *impulse response* $h(k)$ that is the output $y(k)$ when the system is stimulated with an impulsive input $u(k)$

$$u(k) = \begin{cases} 1 & k = 0 \\ 0 & \text{otherwise.} \end{cases}$$

In next lemmas, we are also using the *step response*

$$g(k) = \sum_{i=0}^{k} h(k), \tag{20}$$

and the *ramp response*

$$r(k) = \sum_{i=0}^{k} g(i) \tag{21}$$

of a LTI system.

Thanks to the linear and time-invariance of the system, the output $y(k)$ to any input $u(k)$ is given by the *convolution* of the impulse response $h(k)$ and the input $u(k)$, that is

$$y(k) = h(k) \otimes u(k) = \sum_{i=0}^{k} u(i)h(k-i).$$

With these basic notions recalled, next we state a technical lemma that bounds the output $y(k)$ of a LTI system when the input $u(k)$ belongs to a bounded interval $[\tilde{u} - \bar{\varepsilon}, \tilde{u} + \bar{\varepsilon}]$.

▶ **Lemma 1.** *Given an asymptotically stable discrete-time LTI system with impulse response $h(k)$, step response $g(k)$, input $u(k)$, and output*

$$y(k) = h(k) \otimes u(k).$$

If the input $u(k)$ is bounded as follows

$$u(k) = \tilde{u} + \varepsilon(k), \qquad \tilde{u} \in \mathbb{R},\ -\bar{\varepsilon} \le \varepsilon(k) \le \bar{\varepsilon},$$

then, the output $y(k)$ is bounded by

$$|\tilde{u}| \inf_k \{\text{sign}(\tilde{u})g(k)\} - \bar{\varepsilon}\,\|h\|_1 \le y(k) \le |\tilde{u}| \sup_k \{\text{sign}(\tilde{u})g(k)\} + \bar{\varepsilon}\,\|h\|_1, \tag{22}$$

with the ℓ_1-norm of a signal defined as

$$\|h\|_1 = \sum_{k=0}^{\infty} |h(k)|.$$

Proof. By definition of $y(k)$ as convolution of the impulse response $h(k)$ with the input

signal $u(k)$, it follows

$$\begin{aligned} y(k) &= \sum_{i=0}^{k} u(i)h(k-i) = \sum_{i=0}^{k} (\tilde{u} + \varepsilon(i))h(k-i) \\ &= \tilde{u} \sum_{i=0}^{k} h(k-i) + \sum_{i=0}^{k} \varepsilon(i)h(k-i) \\ &= \tilde{u}\, g(k) + \sum_{i=0}^{k} \varepsilon(i)h(k-i) \\ &\leq |\tilde{u}| \sup_k \{\operatorname{sign}(\tilde{u})g(k)\} + \bar{\varepsilon}\, \|h(k)\|_1 \end{aligned}$$

with

$$\|h(k)\|_1 = \sum_{k=0}^{\infty} |h(k)|.$$

Analogously

$$y(k) \geq |\tilde{u}| \inf_k \{\operatorname{sign}(\tilde{u})g(k)\} - \bar{\varepsilon}\, \|h(k)\|_1,$$

which concludes the proof. ◀

The next Corollary determines the upper and lower bounds to the sum of n consecutive outputs, by exploiting Lemma 1.

▶ **Corollary 1.** *Given an asymptotically stable discrete-time LTI system, if the input $u(k)$ bounded as follows*

$$u(k) = \tilde{u} + \varepsilon(k), \qquad \tilde{u} \in \mathbb{R},\ -\bar{\varepsilon} \leq \varepsilon(k) \leq \bar{\varepsilon}.$$

Then, the sum of n consecutive outputs is bounded by

$$-\big(|\tilde{u}|\, \mathcal{I}(n) + \bar{\varepsilon}\, \mathcal{N}(n)\big) \leq \sum_{k=n_0}^{n_0+n-1} y(k) \leq |\tilde{u}|\, \mathcal{J}(n) + \bar{\varepsilon}\, \mathcal{N}(n), \tag{23}$$

with

$$\mathcal{N}(n) = \sum_{k=0}^{\infty} |g(k) - g(k-n)|, \tag{24}$$

$$\mathcal{I}(n) = \sup_k \{-\operatorname{sign}(\tilde{u})(r(k) - r(k-n))\} \tag{25}$$

$$\mathcal{J}(n) = \sup_k \{\operatorname{sign}(\tilde{u})(r(k) - r(k-n))\} \tag{26}$$

and $g(k)$ and $r(k)$ being the step and ramp response, respectively.

Proof. The output $y(k)$ of a LTI system is the convolution of the impulse response $h(g)$ and the input $u(k)$

$$y(k) = h(k) \otimes u(k).$$

Because of the linearity of the convolution, the sum of n consecutive output is

$$\sum_{i=k}^{k+n-1} y(i) = \left(\sum_{i=k}^{k+n-1} h(i) \right) \otimes u(k) = \big(g(k) - g(k-n)\big) \otimes u(k).$$

Finally, by applying Equation (22) of Lemma 1, Equation (23) of the Corollary follows. ◀

Proof of Theorem 3. Let us first determine the supply function $\mathsf{sbf}_{\text{H}}(t)$ of the HI-criticality server. We aim at modeling the resource supplied to the HI-criticality server as a sequence of active intervals of lengths $S(k)$, interleaved by a sequence of idle intervals of lengths $Z(k)$ that corresponds to the schedule of the LO-criticality server. Formally,

$$S(k) = S_{\text{H}}(k), \qquad Z(k) = S_{\text{L}}(k). \tag{27}$$

In fact, by doing so, Lemma 1 of [22] can give us the supply function of (14) through the proper value of $\sigma_S(n)$ and $\sigma_Z(n)$, as defined in (16).

First of all, the system of (8) that determines the dynamics of $S_{\text{H}}(k)$ is linear. Hence, by the superposition principle the output $S_{\text{H}}(k)$ is equal to the sum of three components:

1. the output $\bar{Q}_{\text{H}}$ when $\varepsilon_{\text{H}}(k) = 0$ and $\varepsilon_{\text{L}}(k) = 0$,
2. the output $y_{\text{HH}}(k)$ when $\bar{Q}_{\text{H}} = 0$ and $\varepsilon_{\text{L}}(k) = 0$, and
3. the output $y_{\text{HL}}(k)$ when $\bar{Q}_{\text{H}} = 0$ and $\varepsilon_{\text{H}}(k) = 0$,

that is

$$S_{\text{H}}(k) = \bar{Q}_{\text{H}} + \underbrace{h_{\text{HH}}(k) \otimes \varepsilon_{\text{H}}(k)}_{y_{\text{HH}}(k)} + \underbrace{h_{\text{HL}}(k) \otimes \varepsilon_{\text{L}}(k)}_{y_{\text{HL}}(k)} \tag{28}$$

and $h_{Hi}(k)$ is the response of $S_{\text{H}}(k)$ to an impulse on the input $\varepsilon_i(k)$, with $i \in \{L, H\}$.

Let us now compute $\sigma_S(n)$ that is, from (16), a lower bound to the sum of the length of n consecutive budgets $S_{\text{H}}(k)$

$$\sigma_S(n) = \inf_{n_0} \sum_{k=n_0}^{n_0+n-1} S(k) = \inf_{n_0} \sum_{k=n_0}^{n_0+n-1} S_{\text{H}}(k) = n\bar{Q}_{\text{H}} + \inf_{n_0} \sum_{k=n_0}^{n_0+n-1} \big(y_{\text{HH}}(k) + y_{\text{HL}}(k)\big). \tag{29}$$

To bound the sum of n consecutive values of $y_{\text{HH}}(k)$ and $y_{\text{HL}}(k)$, we can invoke Corollary 1. Let us start with

$$y_{\text{HH}}(k) = h_{\text{HH}}(k) \otimes \varepsilon_{\text{H}}(k).$$

From the hypothesis of (1), $\varepsilon_{\text{H}}(k)$ is bounded by

$$-\bar{\varepsilon}_{\text{H}} \le \varepsilon_{\text{H}}(k) \le \bar{\varepsilon}_{\text{H}}$$

and then Eq. (23) of Corollary (1) states that

$$-\bar{\varepsilon}_{\text{H}} \mathcal{N}_{\text{HH}}(n) \le \sum_{k=n_0}^{n_0+n-1} y_{\text{HH}}(k),$$

with $\mathcal{N}_{\text{HH}}(n)$ as in (19). Similarly, from the asymmetric bound to $\varepsilon_{\text{L}}(k)$ of (1), from (23) it follows that

$$-\frac{\bar{\varepsilon}_{\text{L}}}{2}(\mathcal{I}_{\text{HL}}(n) + \mathcal{N}_{\text{HL}}(n)) \le \sum_{k=n_0}^{n_0+n-1} y_{\text{HL}}(k),$$

from which the expression of $\sigma_S(n)$ of (17) follows.

The expression of $\sigma_Z(n)$ of (17) can be found by following similar steps:

1. by setting the sequence of idle intervals $Z(k)$ equal to the sequence of the LO-criticality budgets $S_{\text{L}}(k)$, as in (27);

2. by writing the sequence $S_{\text{L}}(k)$ as the sum of $\bar{Q}_{\text{L}}$ and the sequences $y_{\text{LH}}(k)$ and $y_{\text{LL}}(k)$ that corresponds to the responses to the disturbances $\varepsilon_{\text{L}}(k)$ and $\varepsilon_{\text{L}}(k)$ on $S_{\text{L}}(k)$ (similarly as in (28); and
3. by exploiting Corollary 1 to bound $y_{\text{LH}}(k)$ and $y_{\text{LL}}(k)$.

The expressions of $\sigma_{\mathcal{S}}(n)$ and $\sigma_{\mathcal{Z}}(n)$ give the expression of the $\mathsf{sbf}_{\text{H}}(t)$.
Analogously, by setting

$$S(k) = S_{\text{L}}(k), \qquad Z(k) = S_{\text{H}}(k),$$

and following the same steps illustrated above, it is possible to determine the proper values of $\sigma_{\mathcal{S}}(n)$ and $\sigma_{\mathcal{Z}}(n)$ of (18) and then the supply function $\mathsf{sbf}_{\text{L}}(t)$ of the LO-criticality server. This concludes the proof. ◀